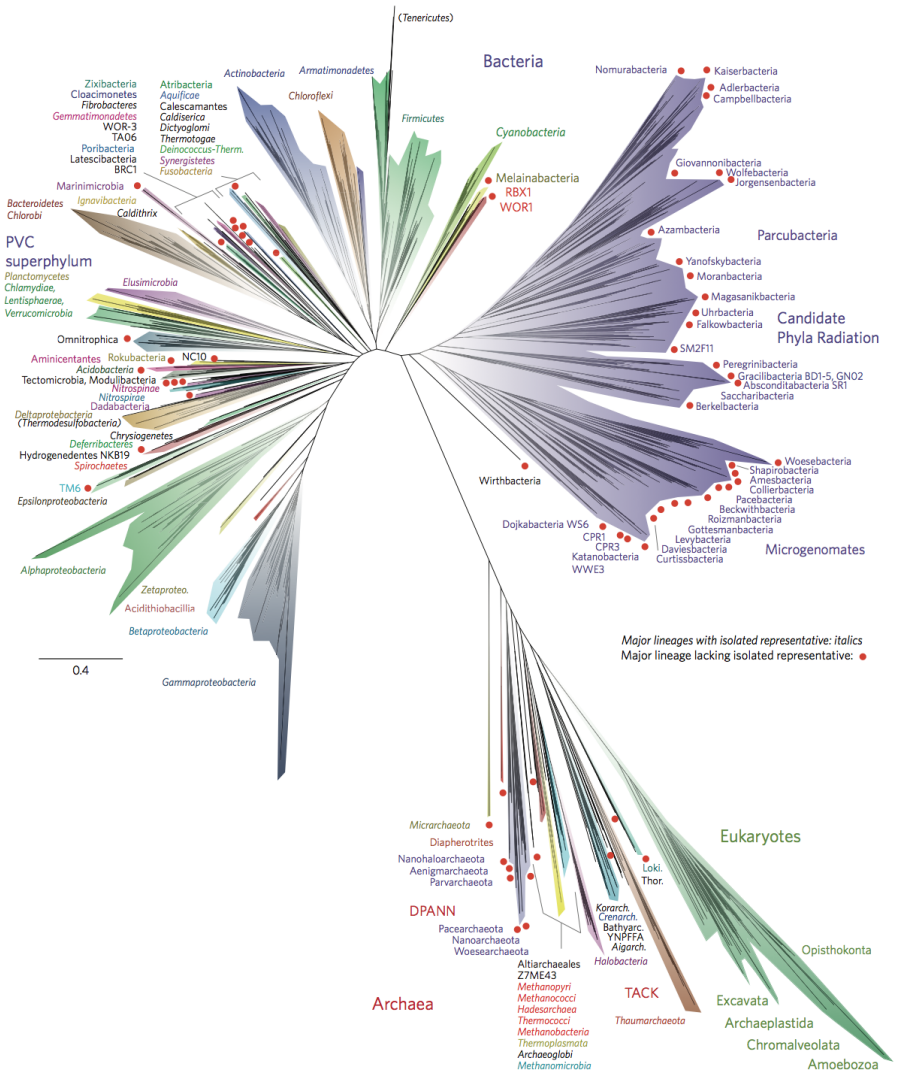
APP3 : Arbres phylogénétiques

Florent Bouchez Tichadou — Gwenaël Delaval
{florent.bouchez-tichadou,gwenael.delaval}@imag.fr

2020—2021



By Laura A. Hug1, Brett J. Baker, Karthik Anantharaman, Christopher T. Brown, Alexander J. Probst, Cindy J. Castelle, Cristina N. Butterfield, Alex W. Hernsdorf, Yuki Amano, Kotaro Ise, Yohey Suzuki, Natasha Dudek, David A. Relman, Kari M. Finstad, Ronald Amundson, Brian C. Thomas and Jillian F. Banfield
http://www.nature.com/articles/nmicrobiol201648

Objectifs d'apprentissage :

— manipuler des données complexes, récursives
— manipuler des structures d'arbres binaires
— écrire des algorithmes récursifs

I. Connaissances à acquérir

À l'issue de cet APP, vous serez capable de :

- écrire des algorithmes permettant de manipuler des arbres binaires : les explorer (hauteur, recherche, etc.) mais également les modifier (ajout ou suppression de nœuds...) ;
- manipuler des structures de données récursives et écrire des algorithmes récursifs ;
- calculer la complexité en temps d'un algorithme opérant sur un arbre binaire, et plus généralement sur une structure de donnée récursive.

II. Organisation des séances d'APP

Cet APP comporte six séances encadrées : 3 séances de groupe, en salle de TD, et 3 séances sur machine.

Entre les séances, du travail personnel est nécessaire et attendu.

Voici un planning indicatif d'avancement.

- Séance Groupe d'ouverture (1h30) : découverte du problème et premières manipulations d'arbre phylogénétique à travers l'acte I. Début (et éventuellement fin) de l'acte II traitant de la recherche d'espèces au sein d'un arbre phylogénétique. Écriture d'algorithmes et calcul de leur complexité.
- Séance Pratique en binômes (3h00) : prise en main des sources founies et implémentation des algorithmes préparées à la Séance Groupe .
- Séance Groupe 2 (1h30) : Terminer l'acte II et travail sur l'acte III.
- Séance Pratique en binômes (3h00) : implémentation des algorithmes vus en Séance Groupe .
- Séance Groupe 3 (1h30) : Fin de l'acte III, avancement possible sur les actes « challenge », écriture d'un algorithme de transformation d'un arbre, calcul de sa complexité. Préparation du tableau d'évaluation.
- Séance Pratique en binômes (3h00) : Fin de l'implémentation des algorithmes.

III. Ressources

- Polycopié du cours : chapitre sur les arbres binaires et sur la récursivité.
- En annexe de ce document les structures de données et les fonctions fournies.

IV. Évaluation

Votre travail en groupe sera évalué formativement à travers d'un tableau présentant un algorithme proposé au sein de cet APP. Vous devez également rendre votre code source : nous vous demandons de fournir un rendu par binômes, sous forme d'archive tar.gz ou zip. Le détail de ce que doit contenir cette archive se trouve sur Caseine, dans l'activité « Rendu de code APP3 ».

V. Rôles

Pour cette séquence APP, inscrivez les personnes volontaires pour chacun des quatre rôles (attention à ne pas reprendre un rôle que vous avez déjà effectué !) :

Modérateur _____

Gardien du temps _____

Scribe _____

Secrétaire _____

VI. APP3 : Arbres phylogénétiques

Situation - problème

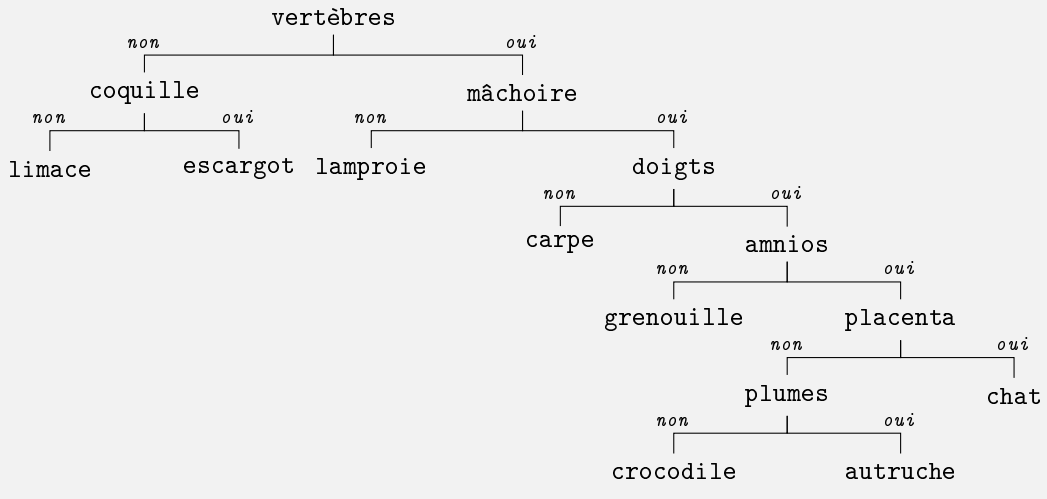
Un *arbre phylogénétique* est un arbre schématique qui montre les relations de parenté entre des groupes d'êtres vivants. Au sein d'un arbre phylogénétique, un groupe d'organismes est défini par une *caractéristique*, commune à tous les organismes de ce groupe. Cette caractéristique, issue de l'évolution, peut être morphologique (par exemple le « pouce opposable » pour le groupe des primates), ou génétique (présence de certains gènes ou protéines).

La construction d'un arbre phylogénétique se fait usuellement à partir d'un tableau de correspondance entre espèces vivantes et caractéristiques. Dans l'exemple suivant, la `lamproie` est listée comme une espèce ne possédant que des `vertèbres` parmi les caractéristiques considérées. Tandis que le `chat` possède toutes les caractéristiques, sauf les `plumes` et la `coquille`.

espèce	amnios	coquille	doigts	mâchoires	placenta	plumes	vertèbres
autruche	oui		oui	oui		oui	oui
carpe				oui			oui
chat	oui		oui	oui	oui		oui
crocodile	oui		oui	oui			oui
escargot		oui					
grenouille			oui	oui			oui
limace							
lamproie							oui

Il y a bien entendu de nombreux arbres phylogénétiques possibles qui correspondent à une table de correspondance donnée. Une des problématiques de la recherche en biologie moderne est de rechercher des arbres phylogénétiques qui correspondent le mieux à une évolution possible des espèces. On peut par exemple considérer que les arbres de hauteur minimale correspondent à l'évolution la plus probable.

La construction d'un arbre phylogénétique à partir d'une table de correspondance peut suivre différentes méthodes et algorithmes. Dans le cadre de cet APP, nous allons considérer les arbres phylogénétiques *binaires* où à la racine on trouve une caractéristique (par exemple `vertèbres`), dans le sous-arbre gauche toutes les espèces qui **n'ont pas** de vertèbres et dans le sous-arbre droit toutes celles qui **en ont**. Voici donc un arbre phylogénétique binaire possible construit à partir de la table de correspondance précédente.



Vous venez de commencer votre stage d'informaticien·ne dans une équipe de recherche en biologie. Cette équipe cherche à développer de nouveaux algorithmes de construction et de modification de la structure d'arbres phylogénétiques. Un·e précédent·e stagiaire a réussi à les convaincre d'utiliser des arbres phylogénétiques binaires, maintenant vos algorithmes vont devoir suivre !

Par exemple, si vous prenez l'arbre précédent, vous ne pourrez pas rajouter simplement l'espèce `lapin` puisqu'elle a toutes les caractéristiques d'un `chat` ! Pour différencier les deux espèces, il faudra rajouter au moins une caractéristique additionnelle (par exemple `canines`).

IMPORTANT L'exemple montre un arbre binaire *entier*, c'est à dire, un arbre où chaque nœud a deux ou zéro fils (si c'est une feuille). Les arbres phylogénétiques que vous serez amenés à construire/manipuler ne seront pas forcément entiers. Par exemple, dans l'arbre ci-dessus, le fait d'enlever la `limace` produit un arbre binaire qui n'est pas entier.

Votre travail consiste à lire un arbre phylogénétique binaire (stocké dans un fichier), et appliquer des analyses sur cet arbre. La fonction de lecture et de construction de l'arbre a été codée par le stagiaire précédent : votre travail commence à partir d'un arbre déjà construit.

Acte I-A

Pour commencer votre travail sur les arbres phylogénétiques, vous vous intéressez aux différentes manières de construire un arbre à partir d'une table de correspondance. Reprenez, dans un premier temps, l'arbre binaire donné précédemment, et transformez le en changeant l'ordre des caractéristiques (en mettant, par exemple, la caractéristique `plumes` à la racine de l'arbre). Quelles observations pouvez-vous faire sur l'arbre binaire résultant ?

Attention, l'arbre doit toujours contenir autant d'information : une espèce présente doit toujours avoir les mêmes caractéristiques que dans l'arbre initial.

Une métrique simple pour pouvoir comparer des arbres phylogénétiques issus du même tableau de caractéristiques est de compter le nombre de caractéristiques : plus ce nombre est faible, moins il y a de duplications dans l'arbre et donc plus élevée est la probabilité que l'arbre corresponde à l'évolution naturelle des espèces.

Vous devez donc en premier lieu écrire un algorithme qui compte le nombre de caractéristiques d'un arbre phylogénétique binaire.

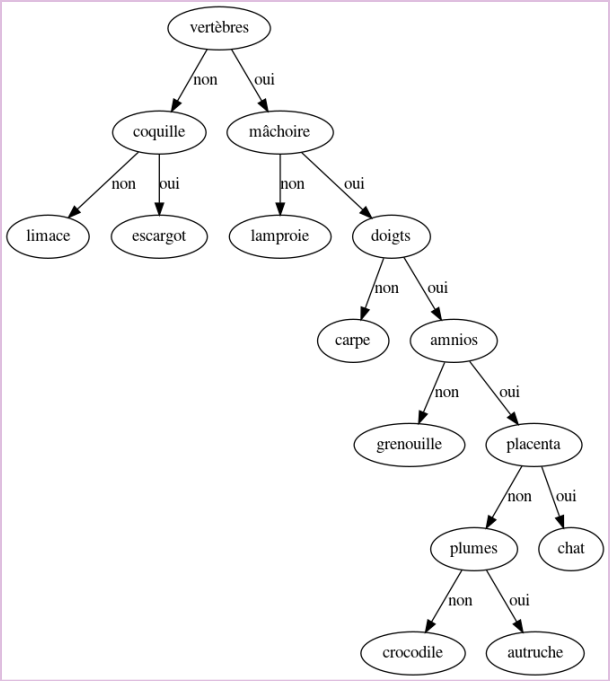
Votre algorithme sera appliqué à des arbres potentiellement très grands (selon les estimations, il existe au total plusieurs millions à plusieurs dizaines de millions d'espèces sur Terre). Il vous est donc demandé de calculer la complexité de votre algorithme et d'ainsi vérifier son applicabilité.

Dans la foulée, modifiez votre algorithme pour qu'il compte, en plus du nombre de caractéristiques, le nombre d'espèces d'un arbre phylogénétique binaire.

Acte I-B (optionnel pour la semaine 1 - nécessaire pour la suite)

Comme tout développeur qui se respecte, vous faites des tests de vos algorithmes dans un ensemble de cas représentatifs (arbre vide, arbre binaire entier, arbre avec uniquement des fils à droite,...). Le problème est que visualiser la structure d'un arbre dans les fichiers avec le format textuel fourni est difficile pour un humain, surtout pour les grands arbres ! Vous décidez d'implémenter des fonctions d'affichage d'arbre pour vous y retrouver.

Nous vous recommandons d'utiliser l'outil GraphViz (https://www.graphviz.org/) qui permet d'obtenir des affichages comme celui présenté ci-dessous à gauche (c'est la visualisation de l'arbre binaire vu précédemment).



Pour obtenir cette image, il faut générer un fichier avec la description de l'arbre en format *dot*. Voici ci-dessous le format *dot* qui correspond à l'arbre de gauche.

```
digraph arbre {
  vertèbres -> coquille [label = "non"]
  coquille -> limace [label = "non"]
  coquille -> escargot [label = "oui"]
  vertèbres -> mâchoire [label = "oui"]
  mâchoire -> lamproie [label = "non"]
  mâchoire -> doigts [label = "oui"]
  doigts -> carpe [label = "non"]
  doigts -> amnios [label = "oui"]
  amnios -> grenouille [label = "non"]
  amnios -> placenta [label = "oui"]
  placenta -> plumes [label = "non"]
  placenta -> chat [label = "oui"]
  plumes -> crocodile [label = "non"]
  plumes -> autruche [label = "oui"]
}
```

Pour créer une image à partir du fichier (`arbre.dot`), par exemple en format `.png`, la commande est la suivante à partir d'une ligne de commande linux :

```
dot -Tpng -o arbre.png arbre.dot
```

Si vous travaillez sous windows, vous trouverez facilement en lignes des sites pour générer une image à partir d'un ficiher `.dot`

Écrire un algorithme qui génère le format GraphViz (`.dot`) d'un arbre phylogénétique binaire. Note : vous pouvez supposer ici qu'il n'y a pas de duplication de caractéristiques.

Acte II

Pour chaque algorithme demandé, l'écrire sous forme algorithmique, calculer sa complexité et l'implémenter.

Nous souhaitons maintenant consulter les caractéristiques des espèces. Écrire un algorithme qui recherche une espèce dans un arbre phylogénétique et affiche toutes ses caractéristiques dans leur ordre d'importance (i.e., une caractéristique plus haut dans l'arbre (plus proche de la racine) doit apparaître avant une caractéristique plus bas dans l'arbre). Si l'espèce n'est pas référencée dans l'arbre, l'algorithme doit explicitement l'indiquer.

Exemple : *Entrée :* "grenouille" *Sortie :* "vertèbres, mâchoire, doigts"

Note : il y a un message caché dans cette partie. Saurez-vous le retrouver ? Indice : les noms de fichiers de tests ont leur importance. . . :-)

Acte III-A

Les scientifiques découvrent chaque année environ 16 000 nouvelles espèces. Il vous faut donc un moyen de rajouter des espèces à vos arbres phylogénétiques.

Écrire un algorithme permettant d'ajouter une espèce associée à un ensemble de caractéristiques dans l'arbre. L'ensemble des caractéristiques ne concerne que les caractéristiques que **possède** l'espèce (et non celles qu'elle ne possède pas). Vous pouvez supposer qu'elles sont données dans l'ordre d'apparition (en descendant) dans l'arbre. Dans le cas où l'espèce possède des caractéristiques ne se trouvant pas encore dans l'arbre, il vous faudra ajouter ces nouvelles caractéristiques aux endroits appropriés. On suppose pour cet acte que lorsqu'une caractéristique n'est pas encore dans l'arbre, l'espèce qu'on est en train d'ajouter est la seule à la posséder - elle est donc absente chez toutes les autres espèces.

Attention : il faut garantir que l'espèce ajoutée est différentiable de toutes les autres espèces présentes, si ce n'est pas le cas, votre programme doit renvoyer l'erreur suivante :
`Ne peut ajouter <espèce>: possède les mêmes caractères que <autre espèce>.`
En revanche, il est possible que l'ajout implique de dupliquer des caractéristiques qui étaient déjà présentes sur d'autres branches. Il n'y a pas de problème à cela.

À ce stade, pour vérifier que votre algorithme fonctionne, il devient nécessaire d'implanter une fonction d'affichage de l'arbre (cf. Acte I-B). Nous vous recommandons également d'écrire une fonction qui écrit l'arbre dans un fichier, en respectant le même format que celui des fichiers de test. Ainsi, vous pourrez sauvegarder vos modifications et relire (réutiliser) vos arbres modifiés avec la fonction de lecture fournie.

Format des fichiers de test

Dans les fichiers de test, les arbres sont représentés à l'aide d'une notation parenthésée.

```
(racine de l'arbre
   (arbre gauche)
   (arbre droit)
)
```

Si l'arbre est vide, il est noté `/`.
Un arbre qui n'a qu'un fils droit serait donc représenté par

```
(racine de l'arbre
   /
   (arbre droit)
)
```

Une feuille est représentée par

```
(feuille)
```

Acte III-B

L'équipe de recherche qui vous emploie souhaite avoir une analyse de la structure de l'arbre : elle a besoin notamment d'avoir la liste des caractéristiques situées **à chaque niveau** de l'arbre afin d'évaluer la pertinence de la classification.

Écrire un algorithme permettant d'afficher cette information niveau par niveau.

Acte IV (difficile)

On souhaite maintenant ajouter une caractéristique à un arbre phylogénétique. La motivation est la suivante : il peut arriver que l'on souhaite ajouter une espèce à un arbre mais que ses caractéristiques coïncident avec celles d'une espèce déjà présente. Dans ce cas, il est nécessaire d'ajouter une nouvelle caractéristique qui permettra de les différencier.

Contrairement à l'acte III, on ne peut plus supposer que les espèces présentes dans l'arbre ne possèdent pas cette nouvelle caractéristique. En conséquence, l'algorithme prendra un arbre, une nouvelle caractéristique, ainsi que l'ensemble des espèces présentes dans l'arbre qui possèdent cette caractéristique. L'arbre doit être mis à jour en insérant la caractéristique à l'endroit approprié. Si plusieurs endroits sont possibles pour l'insertion, la caractéristique devra être insérée à la position la plus proche possible de la racine (ou à la racine si cela est possible).

Attention : Toutes les entrées ne sont pas nécessairement valides. Pour qu'une caractéristique puisse être correctement insérée, il faut que les espèces qui possèdent cette caractéristique forment un clade, c'est-à-dire qu'elle aient un ancêtre commun exclusif. Cela signifie qu'il doit être possible d'insérer un nouveau nœud tel que les descendants de ce nœud soient exactement les espèces listées—et aucune autre. Autrement dit, ces espèces sont les feuilles d'un sous-arbre de l'arbre global.

Pour prendre un exemple, dans l'arbre présenté à l'Acte I, les espèces crocodile, autruche et chat forment bien un sous-arbre et peuvent donc se voir ajouter une caractéristique, par contre crocodile, autruche, chat et carpe n'en forment pas un - il manquerait la grenouille pour disposer d'un sous-arbre complet. Dans le cas où l'ajout d'une caractéristique n'est pas valide, vous devez laisser l'arbre intact et afficher le message d'erreur suivant :
`Ne peut ajouter <caracteristique>: ne forme pas un sous-arbre.`

Acte V (défi)

Écrire un algorithme qui construit un arbre binaire phylogénétique à partir d'une table de correspondance.

Vous devez en particulier définir vous même le format utilisé pour décrire une table de correspondance, ainsi que la manière de lire cette table. Pensez à rajouter ces informations dans le fichier `LISEZMOI.txt` pour que les correcteurs·rices comprennent le travail que vous avez effectué.

Complément d'informations

Vous avez maintenant l'habitude des APPs, vous êtes libres de vous organisez comme vous le souhaitez. Il est important que vous définissiez dans votre groupe un **planning de travail**, pour savoir le temps personnel que vous passerez à lire le cours, réfléchir au problème, etc.

Notez que les actes I, II et III sont **obligatoires** à la fois algorithmiquement, au niveau de l'implantation, mais également au niveau de l'analyse de complexité.

Les actes IV et V sont plus difficiles. Ils soulèvent des questions intéressantes d'un point du vue algorithmique.

Par ailleurs, il y a parsemés dans cet APP plusieurs défis cachés de difficulté croissante. Retrouvez cet défis et validez les exercices correspondants sur AppoLab pour obtenir la note maximale à votre rendu de code... :-)

Vous trouverez pour les TPs dans le répertoire habituel sur Turing et sur Caseine des squelettes de code pour démarrer.

Au besoin, vous pouvez aussi reprendre les fichiers des APPs précédentes (notamment les fichiers `lists.{c/h/py}` si vous avez besoin d'utiliser des séquences). En Python, vous pouvez également utiliser les *listes* fournies de base par le langage.

VII. Exercices d'entraînement

Le chapitre sur les arbres et la récursivité du poly de cours comporte de nombreux exercices. Nous vous recommandons de faire au moins quelques exercices autour de chaque notion importante (voir section suivante) pour vous entraîner.

VIII. Vos apprentissages

Pour chaque objectif de cet APP, estimez vous-même le niveau de vos apprentissages après cette séquence.

Au terme de l'APP, vous êtes capable de :	*Oui*	*Non, mais pour y remédier je vais faire :*
1. Nommer les différents termes et notions autour des arbres : nœuds, racine, feuilles, hauteur d'un arbre ;	☐	
2. Représenter une information sous forme d'arbre binaire ;	☐	
3. Écrire un algorithme récursif simple de parcours et de recherche dans un arbre ;	☐	
4. Représenter par un dessin la pile d'appel des fonctions lors de l'exécution d'un algorithme récursif ;	☐	
5. Expliquer à l'aide d'un dessin l'action d'un algorithme sur un arbre binaire ;	☐	
6. Écrire un algorithme récursif qui modifie un arbre binaire en utilisant la méthode de retour de la racine ;	☐	
7. Calculer la complexité d'un algorithme récursif simple ;	☐	
8. Écrire un algorithme de parcours en largeur d'un arbre binaire en utilisant une structure annexe.	☐	